Verifiability in e-Auction Protocols

Jannik Dreier
Université Grenoble 1, CNRS
jannik.dreier@imag.fr

Hugo Jonker
University of Luxembourg
hugo.jonker@uni.lu

Pascal Lafourcade
Université Grenoble 1, CNRS
pascal.lafourcade@imag.fr

***Abstract*—An electronic auction protocol will only be used by those who trust that it operates correctly. Therefore, e-auction protocols must be *verifiable*: seller, buyer and losing bidders must all be able to determine that the result was correct. We pose that the importance of verifiability for e-auctions necessitates a formal analysis. Consequently, we identify notions of verifiability for each stakeholder. We formalize these and then use the developed framework to perform a detailed study of the verifiability for three examples. We provide an analysis of the protocol by Sako in the applied π-calculus with help of ProVerif, finding it to be correct. Additionally we identify issues with two other protocols.**

I. Introduction

Auctions provide sellers and buyers with a way to exchange goods for a mutually acceptable price. Unlike a marketplace, where the sellers compete with each other, auctions are a seller's market where *buyers* bid against each other over the goods for sale. There are many different types of auctions, varying how to determine winner and price. For example, in an English auction, buyers bid publicly against each other and the highest bid wins (e.g. [1]). A Vickrey auction is rather similar, except that the winning buyer pays the price of the second-highest bid (see e.g. [2], [3]). Conversely, in a Dutch auction, the price starts too high – the auctioneer keeps lowering the price until a buyer claims the good for that price (e.g. [4]). Sealed-bid auctions are auctions that run in one round: in the bidding phase, each buyer submits one sealed bid, which are then simultaneously opened in the opening phase. Sealed-bid auctions can be used to implement auctions where the bidder with the highest bid wins (e.g. [5], [6], [7], [8]), but also Vickrey-style auctions, where the winner pays the second-highest price (e.g. [3]). Finally, goods auctioned in bulk (e.g. flowers) may have more than one winner, each winner paying his own price.

Auctions involve the following stakeholders:

- *Bidders*: parties interested in buying the goods. Their main interest is to acquire the goods at as low a price as possible.
- *Seller*: a party selling goods. His main interest is to sell the goods for as high a price as possible.
- *Auctioneer*: the party organizing the auction. His main interests are to have return customers and high margins (which may depend on the final price of the sold, as done by eBay).

As is readily apparent, the interests of the various stakeholders are opposed. Buyers are in competition with each other for the goods on sale, sellers are in competition with buyers for the price of the goods, the auctioneer may profit directly from overvalued sale price (which provides an incentive to collude with the seller), but a reputation for undervalued sale prices will ensure many repeat customers (which provides incentive to collude with buyers). Consider e.g. the case where there are several bids for the same price. In such a case, an auctioneer might prefer the most "active" bidder instead of the normal tie-breaking rules, and so favor frequent customers over occasional ones.

There are thus no impartial parties to oversee the correctness of the process to determine selling price and winner. For this reason, an auction system must provide some form of *verifiability* for each involved party – irrespective of how the auction process is run and the winner is determined.

Auction verifiability is easy to achieve in isolation, as happens in English auctions. However, maintaining verifiability while ensuring other properties (non-repudiation, privacy, etc.) is far harder. Too often, newly proposed auction protocols proudly show how they achieve these other properties, while only acknowledging the requirement for verifiability in passing. Typically, verifiability is subsequently claimed without providing any formal proof, e.g., [9], [10]. To address this, we propose a generic formal framework applicable independent of the type of auction.

Contribution. The main contribution of this work is to identify a set of scheme-independent definitions, which, taken together, cover verifiability of auctions. To this end, we focus on the bidders (distinguishing verifiability for losing bidders from the winning bidder) and the seller. We present this framework as a set of formal verifiability tests.

Secondly, we use the proposed framework to provide an in-depth analysis of Sako's auction protocol [11]. We prove in ProVerif [12] that the instantiated verifiability tests hold for one or two bids respectively. We then provide a manual reduction to this case for the general case of n bids.

Finally, we investigate the auction protocols by Curtis et al. [10] and by Brandt [9]. Both protocols claim verifiability, yet we identify issues with each.

Related work. There are relatively few formal analyses of auction protocols. Dong et al. [13] study privacy properties of the protocol by Abe and Suzuki [7] in the applied π-calculus. More recently, Dreier et al. [14] used the applied π-calculus to formalize several properties (privacy, non-repudiation, non-cancellation and fairness) for auction protocols, and studied (and found problems with) two auction protocols. Besides these, verifiability in auctions has (to the best of our knowl-

edge) only been studied for particular schemes. However, in the field of voting several more generic definitions of verifiability have emerged. Consequently, we look there for inspiration.

In voting, the property of *individual verifiability* – a voter can verify that her vote counts correctly for the result – has been a well-established notion since the field's inception [15], [16], [17], [18]. Sako and Killian [17] introduced the concept of *universal verifiability*: the property that any observer may verify (using only public information) the correctness of the result. Kremer et al. [19] introduced the notion of *eligibility verifiability*: the property whereby any observer may verify, using only public information, that the set of cast votes from which the result is determined originates only from eligible voters, and each eligible voter cast at most one vote. Finally, Küsters et al. [20] introduced the notion of *accountability*: the property that when verifiability fails, it is possible to identify the person responsible for the failure.

While the intuition behind these notions carries (to some degree) over to auctions, we do note, that unlike voting, auctions involve *competing* parties – hence an illegal bid (e.g., by the seller) may increase the winning price, while not changing the winner. Verification of voting systems thus does not translate directly to verification of auction systems.

Outline. In Section II, we describe our modeling of auctions. In Section III, we formalize the verifiability definitions, taking into account the point of view of the seller and the bidders. In Section IV, we then apply our model to several examples. We prove that Sako's protocol guarantees our verifiability property and we exhibit problems in two other protocols.

II. Modeling Auction Protocols

In this section we describe our model of an auction protocol. We start by explaining the different parts of an auction protocol we need to define verifiability, and then give the formal definition. We only specify the parts necessary to define verifiability to remain as general as possible, yet the model can be interpreted in a more precise framework if needed.

We consider different parties, in particular a set of bidders $\mathcal{B}$ and a seller S. Depending on the protocol there might also be other parties such as an auctioneer, but we do not need to model them for verifiability purposes as only the bidders and the seller verify the execution of the protocol.

Bids are of type Bid (in the simplest case just a price). When being submitted the bids might be encrypted or anonymized to ensure privacy, hence we use the type $EBid$ for such bids. We assume that there is a public list $\mathbb{L}$ of length n and type $List(EBid)$ of all submitted bids, for example a bulletin board. To define the soundness of the verification tests we need a mapping between both types, i.e. a function $getPrice\colon EBid \mapsto Bid$ that gives the bid for an encrypted bid. This function does not need to be computable for any party, as it is only used in the soundness definition.

We also note that the bidders have to register at some point, or are otherwise authenticated when bidding in order to be able to obtain their goods once the auction has ended. This could for example be implemented using signatures, authentication tokens, MACs etc. Therefore we require a function $isReg\colon EBid \mapsto bool$ that returns $true$ if a bid was submitted by a registered bidder, and not modified – this integrity protection is necessary to prevent manipulation of bids.

Finally we require a public function that - given a list of bids - computes the index of the wining bid within the list of all bids: $win\colon List(Bid) \mapsto Index$. This might simply be the index of the maximal bid among all bids, but there may be more complex operations to determine this index depending on the type of auction or to deal with ties (i.e. several maximal bids).

Finally, we assume that the variable $winBid$ of type $Index$ refers to the index of the announced winning bid at the end of the auction, and that each bidder has a variable $myBid$ of type $Index$ that refers to the index of his bid in $\mathbb{L}$.

Note that for a list l we write $l[i]$ to denote the i-th element of the list starting with 1, and $Indices(l)$ to denote the set of indices of l, i.e. $\{1, \ldots, n\}$ if l contains n elements.

Definition 1. *An auction protocol is a tuple* $(\mathcal{B}, S, \mathbb{L}, getPrice, isReg, win, winBid)$ *where*

- $\mathcal{B}$ *is the set of bidders,*
- S *is the seller,*
- $\mathbb{L}$ *is a list of all submitted bids,*
- $getPrice\colon EBid \mapsto Bid$ *is a function that maps submitted bids to bids,*
- $isReg\colon EBid \mapsto \{true, false\}$ *is a function that returns* $true$ *if a bid was submitted by a registered bidder,*
- $win\colon List(Bid) \mapsto Index$ *is a function that returns the index of the winning bid,*
- $winBid$ *is a variable referring to the index of the winning bid at the end of the auction.*

III. Defining Verifiability

In this section, we formally define verifiability for auction protocols. In the first part we consider only first-price auctions. Thereafter we generalise the definitions to account for second-price, multi-price, and other types of auctions.

A. First-Price Auctions

To understand which verifications are needed, we start by discussing three different stakeholder's perspectives:

- A losing bidder wants to be convinced that he actually lost. This can be achieved by proving
  - that the winning bid was actually superior to his bid (as defined by the win function), and
  - that the winning bid was submitted by another bidder (preventing both seller and auctioneer from maliciously adding or manipulating bids to influence the final price).
- A winning bidder wants to check:
  - that he actually submitted the winning bid,
  - that the final price is correctly computed,
  - that all other bids originated from bidders, and
  - that no bid was modified.

Together, these verification checks ensure that the winning bidder is indeed the correct winner, for the correct price. Moreover, the last two checks ensure that the auction process was only influenced by legitimate bidders – neither seller nor auctioneer influenced the process.

- The seller wants to verify that:
  - the announced winner is correct, and
  - the winning price is correct.

in particular if the outcome of the auction was not determined publicly (e.g. privately by the auctioneer, or using distributed computations among the bidders).

To execute these verifications, we introduce some *Verification Tests*.

Definition 2 (Verification Test)**.** *We define a* Verification Test *as an efficient terminating algorithm that takes as input the data visible to a participant of an auction protocol and returns a Boolean value.*

We deliberately do not specify more details at this point as they will depend on the underlying protocol model. Such a test could be a logical formula (whose size is polynomial in the input) in a symbolic model or a polynomial-time Turing-machine in a computational model. Obviously there can be different tests for different participants (e.g. for bidders and the seller), since they may have different views of the protocol execution.

We define verifiability as follows.

Definition 3 (Verifiability - 1st-Price Auctions)**.** *An auction protocol* $(\mathcal{B}, S, \mathbb{L}, getPrice, isReg, win, winBid)$ *ensures* Verifiability *if we have Verification Tests* rv_s, rv_w, ov_l, ov_w, ov_s *respecting the following conditions:*

1) *Soundness:*
   a) *Registration and Integrity Verifiability (RV):*
      - *Anyone can verify that all bids on the list were submitted by registered bidders:* $rv_s = true \implies \forall b \in \mathbb{L}\colon isReg(b) = true$
      - *Anyone can verify that the winning bid is one of the submitted bids:* $rv_w = true \implies winBid \in Indices(\mathbb{L})$
   b) *Outcome Verifiability (OV):*
      - *A losing bidder can verify that his bid was not the winning bid:* $ov_l = true \implies myBid \neq win(getPrice(\mathbb{L}))$
      - *A winning bidder can verify that his bid was the winning bid:* $ov_w = true \implies myBid = win(getPrice(\mathbb{L}))$
      - *The seller can verify that the winning bid is actually the highest submitted bid:* $ov_s = true \implies winBid = win(getPrice(\mathbb{L}))$
2) *Completeness: If all participants follow the protocol correctly, the above tests succeed (i.e., the implications hold in the opposite direction,* $\Longleftarrow$*, as well).*

where – with abuse of notation – we write $getPrice(\mathbb{L})$ *for* $getPrice(\mathbb{L}[1])$, …, $getPrice(\mathbb{L}[n])$.

Consider the perspective of a losing bidder: He can verify that his bid was not the winning bid (ov_l), and that the winning bid was among the ones submitted by registered bidders, which were also not modified (rv_s and rv_w). Similarly a winning bidder can check that his bid was actually the winning bid (ov_w), and that the other bids were submitted by other bidders and not modified (rv_s). Lastly, the seller can also check that the bids using for computing the winner were submitted only by registered bidders (rv_s and rv_w), and that the outcome was correct (ov_s). Hence these tests cover all the verifications discussed above.

Note that in the case of soundness we require the conditions to hold even in the presence of malicious participants (since the tests should check if they did their work correctly), whereas in the case of completeness we only consider honest participants. This is necessary as otherwise e.g. a dishonest auctioneer could announce the correct result, but publish incorrect evidence. Hence the verification tests fail although the outcome is correct, but this acceptable since the auctioneer did not "work correctly" in the sense that he deviated from the protocol specification.

Note that this definition can be applied to sealed-bid auctions, where all bids are submitted in a private way, as well as Dutch or English auctions where offers are publicly announced and the price decreases or increases.

Consider the following toy example: All bidders publish their (not encrypted and not signed) bids on a bulletin board, and at the end of the bidding phase the auctioneer announces the winner. In this case there is a simple test for rv_w: anyone can simply test if the winning bid is one of published ones. However there is no test for rv_s since bids are not authenticated. If we require bidders to sign their bids before publishing them, we also have a simple test for rv_s: verifying the signatures.

It is clear that we have simple tests for ov_l, ov_w and ov_s since everybody can compute the winner on the public list of unencrypted bids. This however means that the protocol ensures no privacy, and no fairness since a bidder can chose his price depending on the previously submitted bids. If we add encryption for the bids to address this shortcomings, the situation becomes more complex and the auctioneer has to prove that he actually computed the winner correctly, for example using zero-knowledge proofs.

B. Other Types of Auctions

To extend our definition to second-price auctions (or more generally $(M+1)$st-price auctions), we have to keep in mind that in this case, the price also depends on the other submitted bids – and not only the winning bid. More generally, we could also imagine situations where the winner has to pay the mean of the first three bids, or other more complex values. Or we can imagine auctions of bulk goods: A seller offers N units of a good, and bidders can make offers such as "I want to buy X units at price Y". In that case there may be a list of winners, and a list of prices they have to pay. To deal with such types of auctions, we generalize our definition as follows.

First, we enrich our model of an auction protocol with a type $Price$. The function win now returns lists of winners and prices $win\colon List(Bid) \mapsto List(Index) \times List(Price)$. We also assume that there are two variables $winPrice$ and $myPrice$ instantiated as the announced list of winning prices and the price announced to a winning bidder respectively. Similarly $winBid$ is now instantiated as a list of indices of bids.

For such auctions, registration verifiability does not change, but winner(s) and seller also want to verify the price they pay to prevent a malicious party from increasing price(s).

Definition 4 (Generalized Verifiability)**.** *An auction protocol* $(\mathcal{B}, S, \mathbb{L}, getPrice, isReg, win, winBid, winPrice)$ *ensures* Verifiability *if we have Verification Tests* rv_s, rv_w, ov_l, ov_w, ov_s *respecting the following conditions:*

1) *Soundness:*
   a) *Registration and Integrity Verifiability (RV):*
      - *Anyone can verify that all bids on the list were submitted by registered bidders:* $rv_s = true \implies \forall b \in \mathbb{L}\colon isReg(b) = true$
      - *Anyone can verify that the winning bids are among the submitted bids:* $rv_w = true \implies \forall b \in winBid\colon b \in Indices(\mathbb{L})$

   b) *Outcome Verifiability (OV):*
   Let $(indexes, prices) = win(getPrice(\mathbb{L}))$
      - *A losing bidder can verify that his bid was not the winning bids:* $ov_l = true \implies myBid \notin indexes$
      - *A winning bidder can verify that his bid was among the winning bids, and that his price is correct:* $ov_w = true \implies \exists i\colon (myBid = indexes[i] \land myPrice = prices[i])$
      - *The seller can verify that the list of winners and the winning prices are correctly determined:* $ov_s = true \implies (winBid = indexes \land winPrice = prices)$
2) *Completeness: If all participants follow the protocol correctly, the above tests succeed (i.e., the implications hold in the opposite direction,* $\impliedby$*, as well).*

where – with abuse of notation – we write $getPrice(\mathbb{L})$ *for* $getPrice(\mathbb{L}[1])$, …, $getPrice(\mathbb{L}[n])$.

Note that e.g. in the case of a second-price auction verifying the price, for example in test ov_w, may implicitly include some more registration verification, namely checking that the second-highest bid was actually submitted by a bidder. Otherwise a malicious seller could add a higher second-highest bid or manipulate the existing one to achieve a higher selling price. This is however included in our model as the function win only works on the list $\mathbb{L}$, hence adding another bid later on to manipulate the bidding price violates the test, and adding or manipulating a bid in $\mathbb{L}$ violates rv_s.

IV. Case Studies

In this section, we discuss our three case studies: The protocols by Sako [11], Curtis et al. [10] and Brandt [9].

A. Protocol by Sako [11]

Kazue Sako [11] proposed a protocol for sealed-bid first-price auction which hides the bids of losing bidders and ensures verifiability. In this protocol dishonest authorities can break privacy, but because of verifiability any manipulation of the auction outcome can be detected.

1) Informal Description: The protocol works as follows:

1) The authorities set up a list of encryption and decryption algorithms E_v and D_v, and a list of constant M_v where each entry corresponds to a bidding price. They publish the encryption algorithms and the constants on a bulletin board.
2) To bid for price p, a bidder encrypts M_p using E_p, signs it and publishes the bid $C_p = E_p(M_p)$ together with the signature on the bulletin board.
3) After the bidding phase is over, the authorities check the signatures and start decrypting with the highest possible price $t = p_{max}$. If $D_t(C_i) = M_t$, then the bid i was a bid for price t. If all decryptions fail, the authorities decrease t and try again. Each time a decryption is done, they publish a proof of correct decryption to enable verifiability. This can be a zero-knowledge proof, or it might be achieved by simply publishing the secret key as in one of the examples in the original paper [11].
4) To verify the outcome, anybody can verify the signatures, and check the proofs of correct decryption.

2) Formal Model: We formalize this protocol using a set of bidders $\mathcal{B}$ and a seller S. The list of all submitted bids $\mathbb{L}$ is published on the bulletin board. The function $getPrice(C)$ decrypts the bid by trying all possible prices until the correct value is found, i.e. until $D_t(C) = M_t$ (as the authorities would), and then returns t. The function $isReg$ simply checks the signature. The function win returns the index of the highest bid, and $winBid$ will point to the index of the winning bid at the end of the auction as announced by the authorities on the bulletin board.

3) Analysis summary: In this part, we analyze verifiability given the above model and protocol description. We start with a high-level summary, then model the tests in the applied π-calculus and claim their soundness and completeness for this protocol (proofs of which are available in the techreport [**?**]).

a) Summary: In the original paper Sako sketches some verifications, which can be translated to our model as follows. The test rv_s simply checks all the signatures. For rv_w one can check if the encrypted value appears in the list of bids on the bulletin board when the winner is announced. Finally the test for ov_l, ov_w and ov_s works as follows: Any participant can check that all decryptions corresponding to a potentially higher bid were unsuccessful (i.e. the result was different from M_t), and verify the proofs of correct decryption. To check if he won or lost, a bidder can simply compare his bid to the

winning price. Similarly the seller can check if the announced winning bid is actually the winning bid.

We can easily model the verifiability tests as processes as well, they will simply receive the necessary data on some channels and output a certain message depending on whether they accepted this input or not. This also allows us to check soundness and completeness of the tests in ProVerif, since these properties can be modeled as reachability properties such as "If the input to the test was generated by honest participants according to the protocol, can the process "test" emit a message "KO"?" (which corresponds to completeness) or "Is it possible for an attacker to send messages to the process "test" such that it emits a message "OK" although the input is not correct?" (which corresponds to soundness).

The first test rv_s, described in Listing 1, is actually a simple implementation of our soundness condition ($\forall b \in \mathbb{L} : isReg(b) = true$): It receives all bidders' public keys and the bids, and then verifies all signatures. We show that is sound and complete in Theorem 1.

```
1 let testrvs =
2  in(chK1,k1); ... in(chKn,kn);
3  in(chBB1,(m1,s1)); ... in(chBBn,(mn,sn));
4  if checksign(s1,k1) = m1 && ... &&
5   checksign(sn,kn) = mn then out(chRVS,OK)
6  else out(chRVS,KO).
```

Listing 1. The test rv_s.

Theorem 1. *The test rv_s (see Listing 1) for the protocol by Sako [11] is sound and complete.*

Proof. See the tech report [**?**]. □

The second test rv_w, given in Listing 2, takes all published bids on channel `chBB` and the winning bid published on channel `chW` and checks if the winning bid is among the published ones. This is again the direct implementation of the soundness condition, and we show that it is sound and complete in the following theorem.

```
1 let testrvw =
2  in(chBB1,(m1,s1)); ... in(chBBn,(mn,sn));
3  in(chW,(m,ind,price));
4  if m1=m || ... || mn=m then
5   out(chRVW,OK)
6  else
7   out(chRVW,KO).
```

Listing 2. The test rv_w.

Theorem 2. *The test rv_w (see Listing 2) for the protocol by Sako [11] is sound and complete.*

Proof. See the tech report [**?**]. □

For the outcome verification tests we employ a similar approach. Note that for Sako's protocol the tests ov_l, ov_w and ov_s are all the same, and are described by process `testov`. Due to space considerations, we refer to the tech report for the listing of the test.

Theorem 3. *The test `testov` for the protocol by Sako [11] is sound and complete.*

Proof. See the tech report [**?**]. □

We thus conclude that all proposed tests are sound and complete, hence the protocol by Sako [11] is *verifiable*[1]

B. Protocol by Curtis et al. [10]

The protocol by Curtis et al. [10] was designed to support any type of sealed-bid auction while guaranteeing fairness, privacy, verifiability and non-repudiation.

1) Informal Description: The main idea of the protocol is the following: The bidders register with a trusted Registration Authority (RA) using a Public-Key Infrastructure (PKI), which issues pseudonyms that will then be used for submitting bids to the Seller (S). The seller eventually receives all bids in clear and can hence apply any auction function possible, yet he cannot link a bid to a bidder because of the pseudonyms. The protocol is split into three phases: Registration, Bidding, and Winner determination.

- *Registration:* Each bidder sends his identity, a hash of his bidding price b_i and a signature of $h(b_i)$ to the RA. The RA checks the identity and the signature using the PKI, and replies with an encrypted and signed message containing a newly generated pseudonym p and the hashed bid $h(b_i)$.
- *Bidding:* The RA generates a new symmetric key k. Each bidder will send $c = Enc_{pk_S}(b_i)$, his bid b_i encrypted with the seller's public key, and a signature of c, together with his pseudonym to the RA. The RA will reply with a signature on c, and encrypts the bidders message, together with the hashed bid $h(b_i)$ from phase one, using the symmetric key k. This encrypted message is then send to the seller.
- *Winner determination:* After all bids have been submitted, the RA will reveal the symmetric key k to the seller. The seller can then decrypt the bids, verify the correctness of the hash and determine the winner. To identify the winner using the pseudonym he can ask the RA to reveal the true identity.

2) Formal Model: We have the set of bidders $\mathcal{B}$ and a seller S. We do not need to specify the type of bids Bid since the protocol supports any type of bids. The bids are published when the auctioneer reveals the symmetric key, i.e. $\mathbb{L}$ contains bids of the following type: $(Pseudo \times PEnc(Bid) \times Hash)$, where $Pseudo$ is the type of pseudonyms, $PEnc$ is a public-key encryption and $Hash$ are hash values. The function $getPrice$ will simply decrypt the encrypted bid (the second entry of the tuple). The function $isReg$ will return $true$ if and

[1] The ProVerif input files for Sako's protocol are available from http://www-verimag.imag.fr/~dreier/papers/csf-code.zip as part of a detailed case study of multiple protocols.

only if the hash value is correct, the pseudonym was actually attributed by the RA and the bid was submitted correctly signed by the bidder with this pseudonym. The protocol is independent of the used auction mechanism and hence does not define win. The seller will simply decrypt all bids and can then apply any function win. He will publish the winning price and the winning bidders pseudonym, and $winBid$ will denote the index of the bid containing this pseudonym.

3) Analysis: Since the seller does the winner determination on his own, there is a simple test for ov_s: He can check his own computations. As the computation of the winner is not specified in order to support any type of auction, we cannot give tests for ov_l and ov_w – they would have to be designed as a function of the used auction algorithm. Yet there is also a test for rv_w: Checking if the pseudonym appears in the list of bids.

However, the messages from the RA to the seller are not authenticated, hence there can be no suitable tests for rv_s once the (encrypted) bids are revealed. Even if they were authenticated, this still requires trusting the RA (contrary to the goal of verifiability) since there is no way to verify if a pseudonym actually corresponds to a bidder. This also shows a simple attack: the RA can create a new pseudonym and submit a bid under this pseudonym, which may allow him to manipulate the auction outcome. protocol provides no test for rv_w and ov_w since the seller is the only one capable of decrypting the bids and computing the winner.

C. Protocol by Brandt [9]

The protocol by Brandt [9] realizes a first-price sealed-bid auction and was designed to ensure full privacy in a completely distributed way. It exploits the homomorphic properties of a distributed El-Gamal Encryption scheme for a secure multi-party computation of the winner.

1) Informal Description: The participating bidders and the seller communicate using a bulletin board, i.e. an append-only memory accessible for everybody. The bids are encoded as bit-vectors where each entry corresponds to a price. The protocol then uses linear algebra operations on the bid vectors to compute a function f_i, which returns a vector containing one entry "1" if the bidder i submitted the highest bid, and different numbers ($\neq 1$) otherwise. To be able to compute this function in a completely distributed way, and to guarantee that no coalition of malicious bidders can break privacy, these computations are performed on the encrypted bids using homomorphic properties of a distributed El-Gamal Encryption.

In a nutshell, the protocol realizes the following steps:

1) Firstly, the distributed key is generated: each bidder chooses his part of the secret key and publishes the corresponding part of the public key on the bulletin board.
2) Each bidder then computes the joint public key, encrypts his offer using this key and publishes it on the bulletin board.
3) Then the auction function f_i is calculated for every bidder using some operations exploiting the homomorphic property of the encryption scheme.
4) The outcome of this computation (n encrypted values) are published on the bulletin board, and each bidder partly decrypts each value using his secret key.
5) These shares are send to the seller, who can combine all to obtain the result (i.e. all f_i). He publishes part of the shares such that each bidder j can only compute his f_j to see if he won or lost (using his knowledge and the published shares), but not the other f_i.

2) Formal Model: We have a set of bidders $\mathcal{B}$ and a seller S. The list of all submitted bids $\mathbb{L}$ is published on the bulletin board. The function $getPrice(C)$ decrypts the bid using the joint private key. The function win returns the index of the highest bid submitted, in case of ties the one submitted by the bidder with the smallest index. The protocol has two particularities: Firstly there is no registration (and hence no meaningful function $isReg$), and secondly the winner is not publicly announced – only the winning bidder and the seller know at the end who won. We can still assume that $winBid$ gives the index of the winning bid, although only the seller and the winning bidder have access to it. We assume that there is a magical function $isReg$ that can check if a bid was submitted by a registered bidder, however the absence of registration and authentication means that we cannot implement it.

3) Analysis: The protocol includes no authentication or registration, hence there is no suitable test for rv_s. An attacker may hence submit bids on behalf of a bidder, which cannot be detected using a verification test. Yet using the values published on the bulletin board everybody can check if the values used for the computation were the previously submitted bids, and as the winning index will be among them, we have a test for rv_w.

The author claims that the protocol is verifiable as the parties have to provide zero-knowledge proofs for their computations, however there are two problems.

Firstly a winning bidder cannot verify if he actually won. To achieve privacy, the protocol hides all outputs of f_i except for the entry containing "1". This is done by exponentiation of all entries x_i of the return vector x with random values, i.e. by calculating $x_i^{\sum_j r_j}$. If x_i is one, this will still return one, but a random value for any other value of x. Yet these random values r_j may add up to zero (mod q), hence the returned value will be $x_i^0 = 1$ and the bidder will conclude that he won ($x_i = 1$), although he actually lost ($x_i \neq 1$). Hence simply verifying the proofs is not sufficient – such a test ov_w would not be sound. For the same reason the seller might observe two or more "1"-values even though all proofs are correct, and will be unable to decide which bidder actually won. He could even exploit such a situation to his advantage: He can simply tell both bidders that they won and take money from both, although there is only one good to sell. If the bidders do not exchange additional data there is no way for them to discover that something went wrong, since the seller is the

only party having access to all values. The probability of the random values adding up to zero is low, yet this means that there are cases where the verifiability tests are not sound.

Secondly the paper does not exactly specify the proofs that have to be provided in the joint decryption phase. If the bidders only prove that they use the same private key on all decryptions (and not also that it is the one they used to generate their public key), they may use a wrong one. This will lead to a wrong decryption where with very high probability no value is "1", as they will be random. Hence all bidders will think that they lost, thus allowing a malicious bidder to block the whole auction, as no winner is determined. Hence, if we assume that ov_l consists in verifying the proofs, a bidder trying to verify that he lost using the proofs might perform the verification successfully, although the result is incorrect and he actually won – since he would have observed a "1" if the vector had been correctly decrypted. This problem can be addressed by requiring the bidders to also prove that they used the same private key as in the key generation phase.

V. Conclusion

In this work, we identified the types of verifiability necessary for the stakeholders in auctions. We then formalized these requirements in a protocol-independent way, resulting in tests $rv_s, rv_w, ov_l, ov_w, ov_s$, which together constitute a general verifiability framework for auction protocols.

We illustrated the use of the proposed tests by three case studies. In the first case study, we provided an in-depth analysis of the protocol by Sako [11], the first work to fully hide the bids of losing bidders. For this, we formalized Sako's protocol in the applied π-calculus, and proved soundness and completeness of the instantiation of our tests for this protocol. The case for one (resp. two) bid(s) was proven using ProVerif, the case for n bids was reduced to this case by manual proof. In addition, we analyzed the auction protocols by Curtis et al. [10] and by Brandt [9]. The protocol by Curtis et al. is correct only for a trusted Registration authority – which runs contrary to the point of verification: that the authorities no longer need to be trusted. Brandt's protocol does not have sound verifiability tests: it is technically possible for a losing bidder to conclude he won. Moreover, it may also be possible for a bidder to prevent anyone from winning by using a wrong decryption key. To prevent this, bidders must prove that the private key matches the previously announced public key.

Future work. The proposed tests enable protocol independent reasoning about auction verifiability. The developed tests have been applied *a posteriori*, which works well for sealed-bid auctions. We are investigating how to extend the framework to handle interactive auctions, such as traditional English auctions. In such auctions a non-bidder may submit a higher bid. An interested bidder may then overbid, before realizing that this was not necessary. Checks executed after the auction will not prevent this. Looking further ahead, we are interested in the full relationship between fairness and verifiability in auctions. As illustrated, there exist verifiability requirements without which violations of fairness may occur. The exact relationship between fairness and verifiability however is an open question.

References

[1] K. Omote and A. Miyaji, "A practical english auction with one-time registration," in *Proc. 6th Australasian Conference on Information Security and Privacy*, ser. LNCS, vol. 2119. Springer, 2001, pp. 221–234.

[2] M. Harkavy, J. D. Tygar, and H. Kikuchi, "Electronic auctions with private bids," in *Proc. 3rd USENIX Workshop on Electronic Commerce*. Usenix, 1998.

[3] H. Lipmaa, N. Asokan, and V. Niemi, "Secure vickrey auctions without threshold trust," in *Proc. 6th Conference on Financial Cryptography*, ser. LNCS, vol. 2357. Springer, 2003, pp. 87–101.

[4] T. E. Rockoff and M. Groves, "Design of an internet-based system for remote dutch auctions," *Internet Research*, vol. 5, pp. 10–16, 1995.

[5] C. Cachin, "Efficient private bidding and auctions with an oblivious third party," in *Proc. 6th Conference on Computer and Communications Security (CCS'99)*. ACM Press, 1999, pp. 120–127.

[6] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proc. 1st Conference on Electronic Commerce*. ACM Press, 1999, pp. 129–139.

[7] M. Abe and K. Suzuki, "Receipt-free sealed-bid auction," in *Proc. 5th Conference on Information Security*, ser. LNCS, vol. 2433. Springer, 2002, pp. 191–199.

[8] X. Chen, B. Lee, and K. Kim, "Receipt-free electronic auction schemes using homomorphic encryption," in *Proc. 6th Conference on Information Security and Cryptology*, ser. LNCS, vol. 2971. Springer, 2003, pp. 259–273.

[9] F. Brandt, "How to obtain full privacy in auctions," *International Journal of Information Security*, vol. 5, pp. 201–216, 2006.

[10] B. Curtis, J. Pieprzyk, and J. Seruga, "An efficient eAuction protocol," in *Proc. 7th Conference on Availability, Reliability and Security (ARES'07)*. IEEE Computer Society, 2007, pp. 417–421.

[11] K. Sako, "An auction protocol which hides bids of losers," in *Proc. 3rd Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000)*, ser. LNCS, H. Imai and Y. Zheng, Eds., vol. 1751. Springer, 2000, pp. 422–432.

[12] B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules," in *Proc. 14th Computer Security Foundations Workshop (CSFW-14)*. Cape Breton, Nova Scotia, Canada: IEEE Computer Society, Jun. 2001, pp. 82–96.

[13] N. Dong, H. L. Jonker, and J. Pang, "Analysis of a receipt-free auction protocol in the applied pi calculus," in *Proc. 7th Workshop on Formal Aspects in Security and Trust (FAST'10)*, ser. LNCS, vol. 6561. Springer-Verlag, 2011, pp. 223–238.

[14] J. Dreier, P. Lafourcade, and Y. Lakhnech, "Formal verification of e-auction protocols," Verimag, Tech. Rep. TR-2012-17, 2012.

[15] A. Fujioka, T. Okamoto, and K. Ohta, "A practical secret voting scheme for large scale elections," in *Advances in Cryptology – AUSCRYPT '92*, ser. LNCS, J. Seberry and Y. Zheng, Eds. Springer Berlin / Heidelberg, 1992, vol. 718, pp. 244–251.

[16] J. Benaloh and D. Tuinstra, "Receipt-free secret-ballot elections (extended abstract)," in *Proc. 26th Symposium on Theory of Computing*, ser. STOC '94. New York, NY, USA: ACM, 1994, pp. 544–553.

[17] K. Sako and J. Kilian, "Receipt-free mix-type voting scheme - A practicalsolution to the implementation of a voting booth," in *Proc. 14th Conference on the Theory andApplications of Cryptographic Techniques: Advances inCryptology (EUROCRYPT'95)*, ser. LNCS, vol. 921. Springer, 1995, pp. 393–403.

[18] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *Proc. 19th Annual Conference on the Theory andApplications of Cryptographic Techniques: Advances inCryptology (EUROCRYPT)*, ser. LNCS, vol. 1807. Springer, 2000, pp. 539–556.

[19] S. Kremer, M. D. Ryan, and B. Smyth, "Election verifiability in electronic voting protocols," in *Proc. 15th European Symposium on Research in Computer Security (ESORICS 2010)*, ser. LNCS, vol. 6345. Springer, 2010, pp. 389–404.

[20] R. Küsters, T. Truderung, and A. Vogt, "Accountability: definition and relationship to verifiability," in *Proc. 17th Conference on Computer and Communications Security (CCS'10)*, ser. CCS '10. ACM, 2010, pp. 526–535.